



Eclipse and Embedded Linux Developers: What It Can and Cannot Do for You

Anna Dushistova

anna_dushistova@mentor.com

Comprehensive Solutions for

Android™ ▪ Nucleus® ▪ Linux®

Mobile & Beyond ▪ 2D/3D User Interfaces ▪ Multi-OS ▪ Networking



Android is a trademark of Google Inc. Use of this trademark is subject to Google Permissions.

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

Why IDE?

■ From wikipedia:

- “...IDEs are designed to maximize programmer productivity by providing **tightly-knit components** with **similar user interfaces**. This should mean that the programmer has **much less mode switching** to do than when using discrete development programs...”

Typical Content of an IDE

- a source code editor
- a compiler and/or an interpreter
- build automation tools
- a debugger
- SCM integration

About Eclipse

■ What is it?

- Eclipse is an open source community, whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle.

■ History of Eclipse

- Started in November 2001 by Borland, IBM, MERANT, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft and Webgain
- On Feb 2, 2004 the Eclipse Board of Stewards announced Eclipse's reorganization into a not-for-profit corporation. The founding Strategic Developers and Strategic Consumers were Ericsson, HP, IBM, Intel, MontaVista Software, QNX, SAP and Serena Software.

What can Eclipse offer?

- C/C++ Development Tooling(CDT) for edit/compile/debug cycle
- Target Management (TM) for working with remote hosts
- CVS, SVN and Git integration
- Linux Tools for different Linux specific tools (autotools, oprofile, lttnng, etc.)

Some companies behind these projects



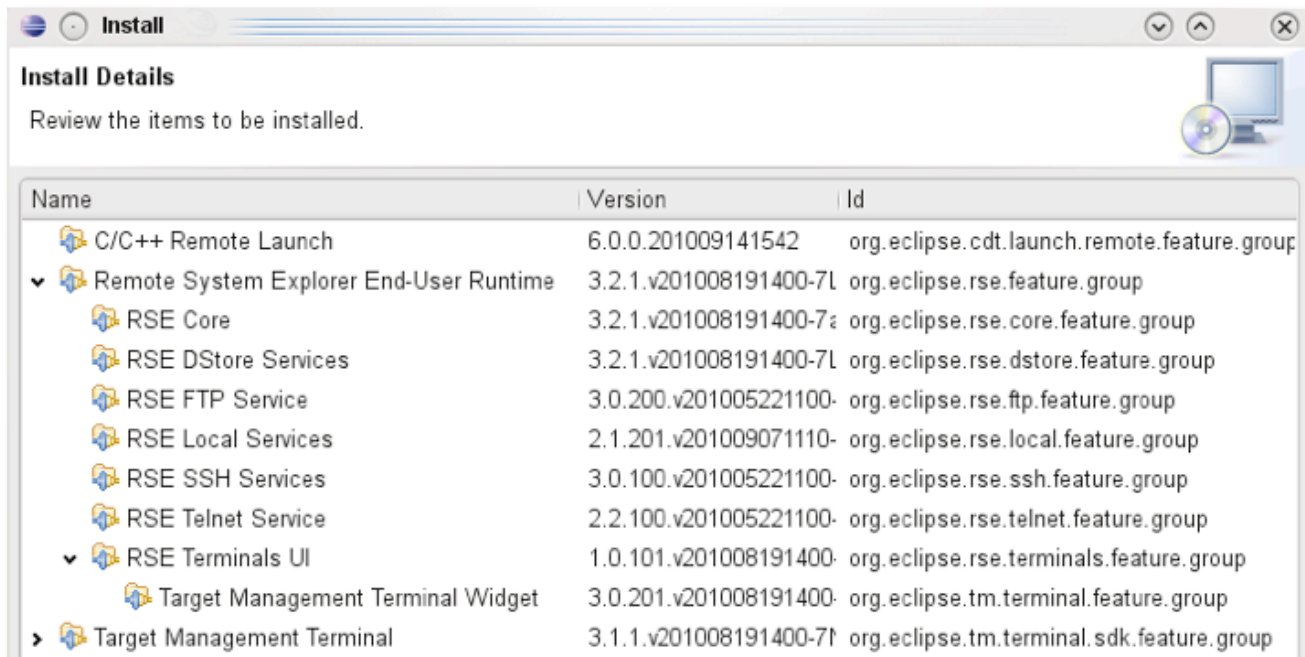
Comprehensive Solutions for

Android ▪ Nucleus ▪ Linux

Mobile & Beyond ▪ 2D/3D User Interfaces ▪ Multi-OS ▪ Networking

But...

- Is it possible to just use Eclipse with these plugins as is for embedded Linux development?
 - Let's start with Eclipse IDE for C/C++ Linux Developers package (<http://www.eclipse.org/downloads/packages/eclipse-ide-cc-linux-developers-incubating-components/heliossr1>)
 - Install missing Target Management features



Comprehensive Solutions for

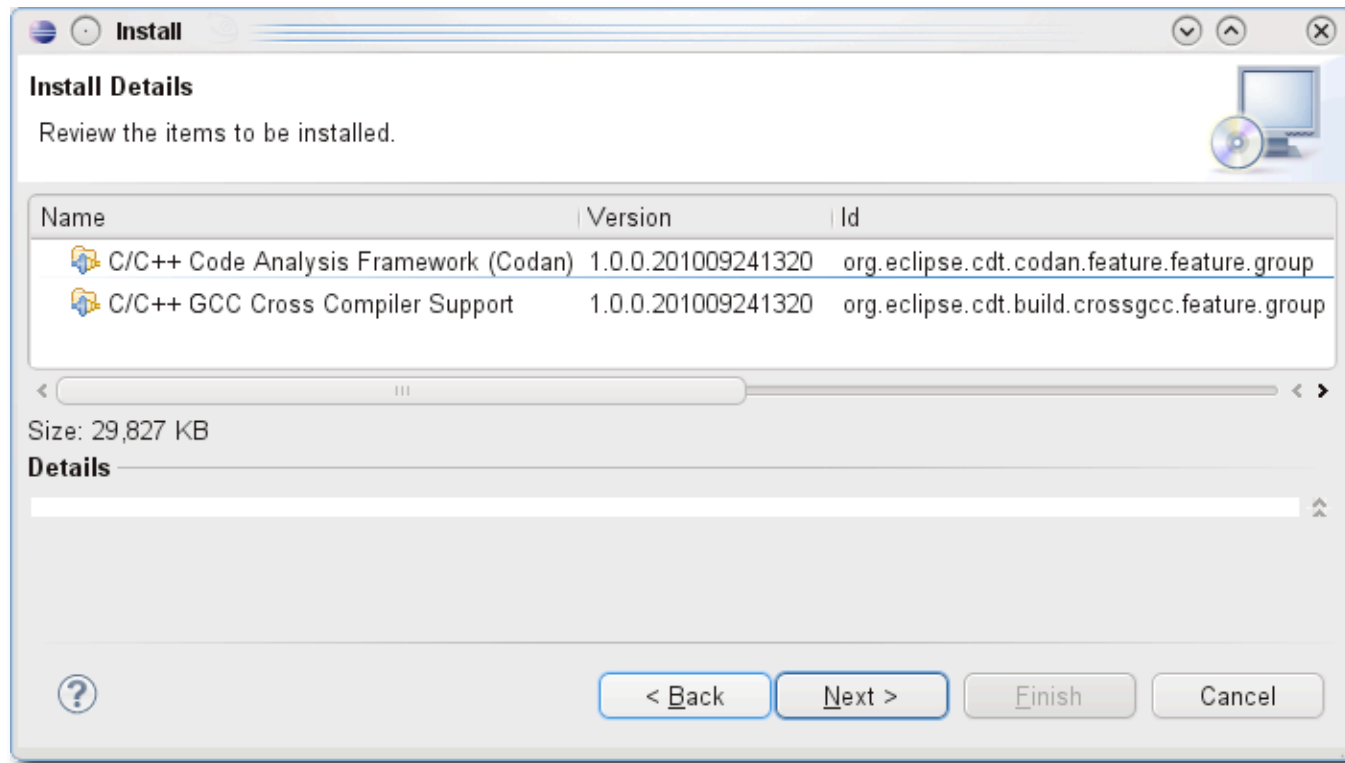
Android ■ Nucleus ■ Linux

Mobile & Beyond ■ 2D/3D User Interfaces ■ Multi-OS ■ Networking

Install additional CDT features

■ Not everything we need is in that package

- Download <http://www.eclipse.org/downloads/download.php?file=/tools/cdt/releases/helios/dist/cdt-master-7.0.1-I201009241320.zip>
- Install Codan and cross compiler support

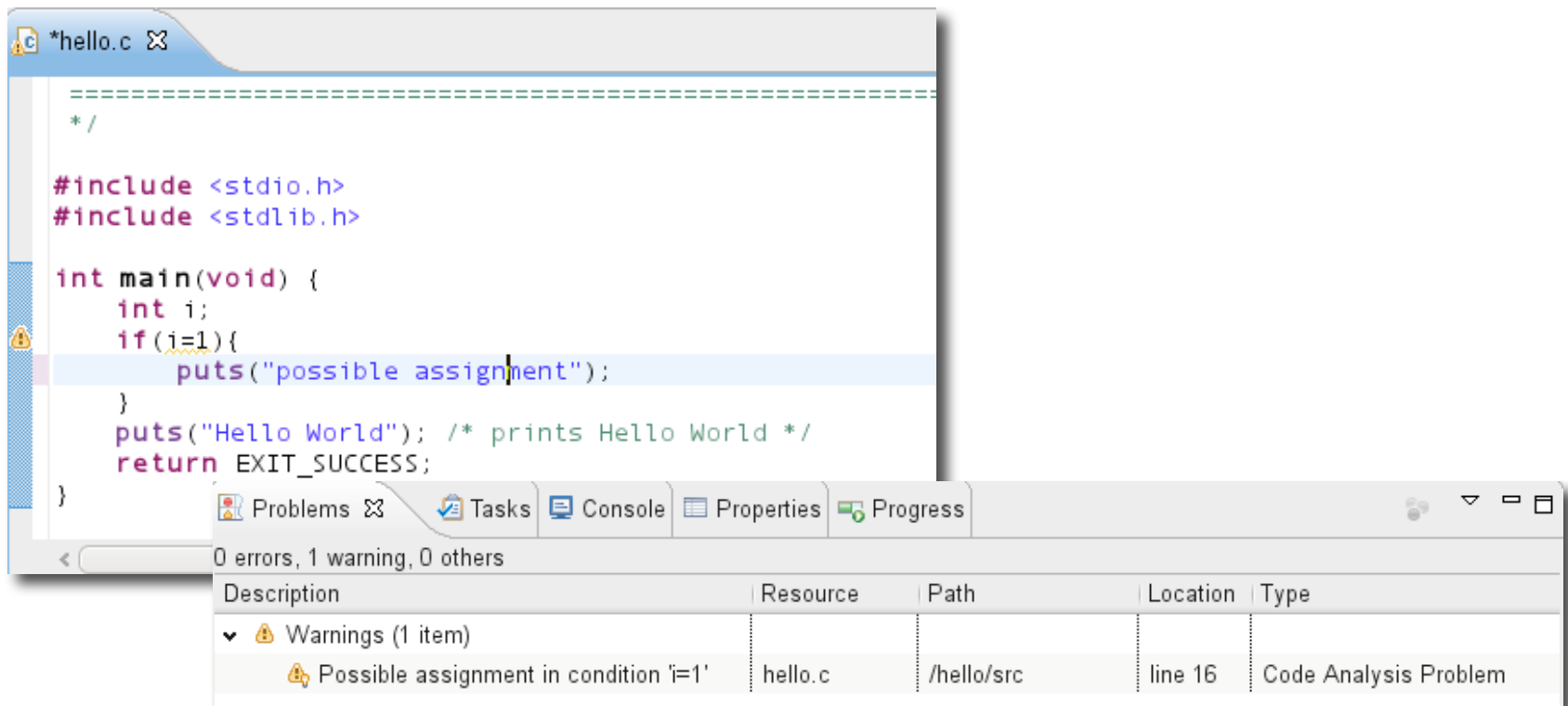


Working with C/C++ code

■ New in CDT 7.0:

– Code Analysis Framework “Codan”

- For additional information see <http://wiki.eclipse.org/CDT/designs/StaticAnalysis>



Building your application

- Works only for a local toolchain out of the box!
- How can we integrate CDT with our cross compilation tools?
 - Write a plugin for Eclipse that adds support for it
 - Try using “Cross GCC”
 - Redefine settings for each project manually

Defining your own toolchain

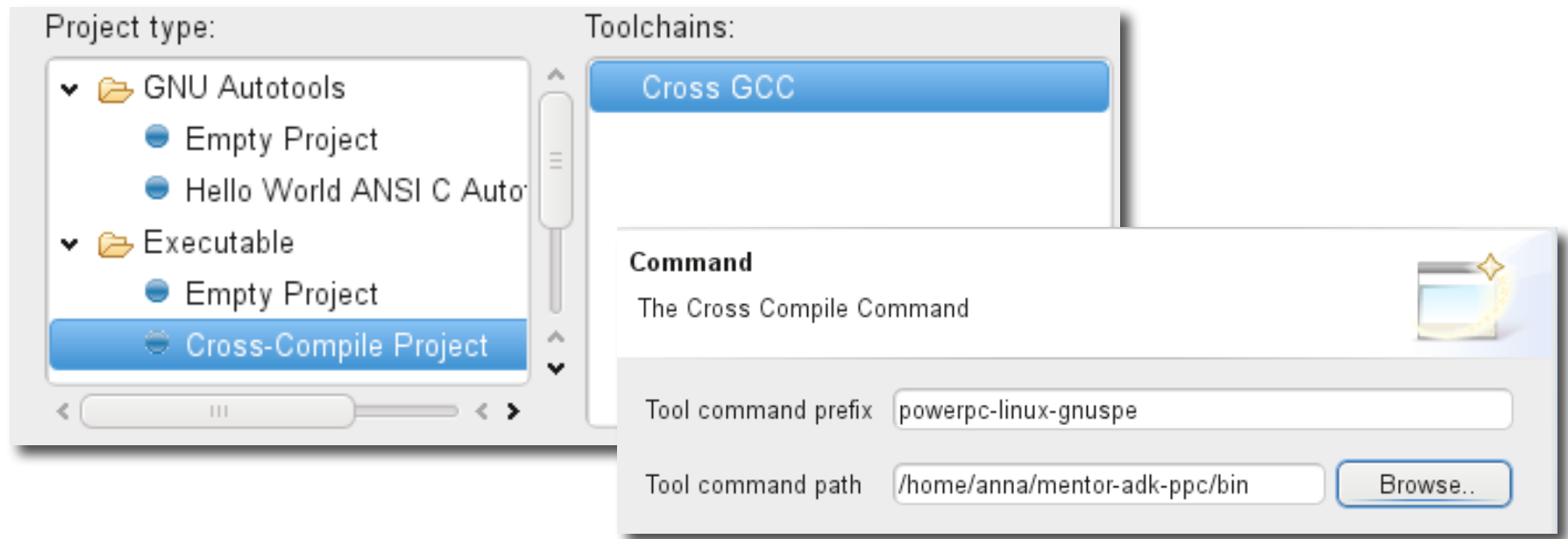
- If you don't need any special settings, just extend existing definitions for "Linux GCC"
- The more special your needs are, the more changes you'll have to make
- Changes are not obvious and require some Eclipse experience

```
name="Anna's GCC" osList="all">
<targetPlatform
    id="cdttest309024.gnu.platform.base"
    name="Debug"
    binaryParser="org.eclipse.cdt.core.ELF"
    osList="linux,hpux,aix,qnx"
    archList="all">
</targetPlatform>
<builder
    superClass="cdt.managedbuild.target.gnu.builder"
    id="cdt.test309024.builder.base">
</builder>

<tool command="$(CXX)"
    id="cdt.test309024.cpp.compiler" isAbstract="false"
    name="Anna's GNU G++ Compiler" superClass="cdt.managedbuild.tool.gnu.cpp.compiler">
</tool>
```

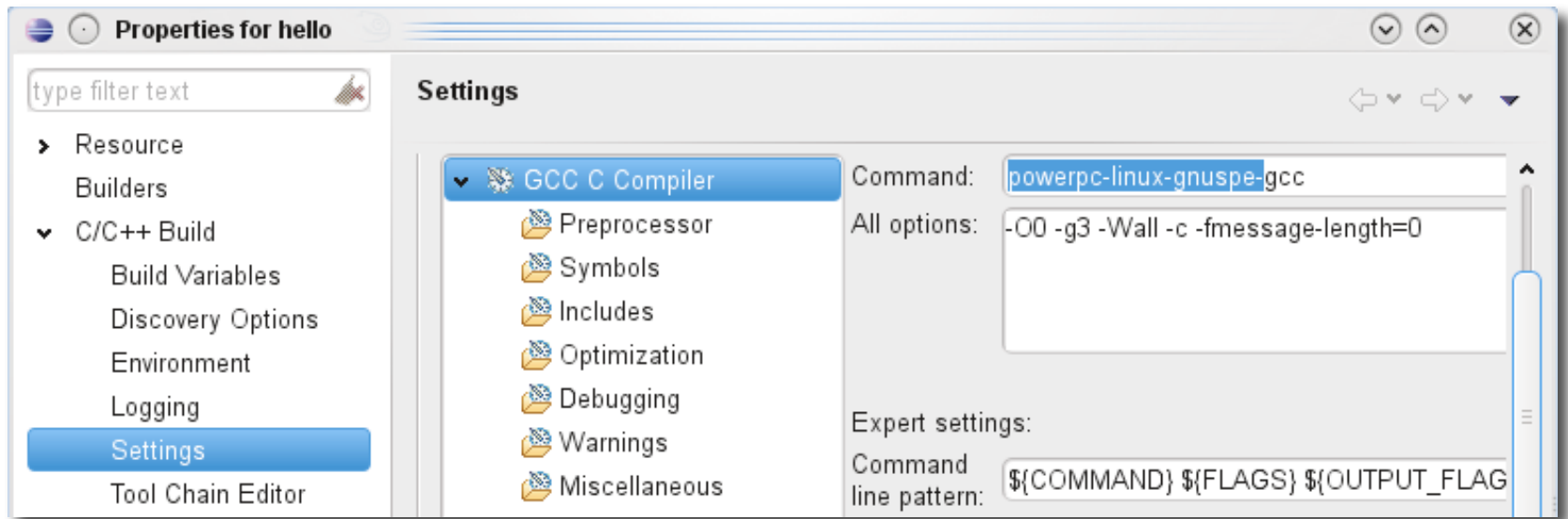
Using “Cross GCC”

- Asks you for just two settings:
 - path
 - prefix(make sure you add dash in the end)
- Not quite there yet



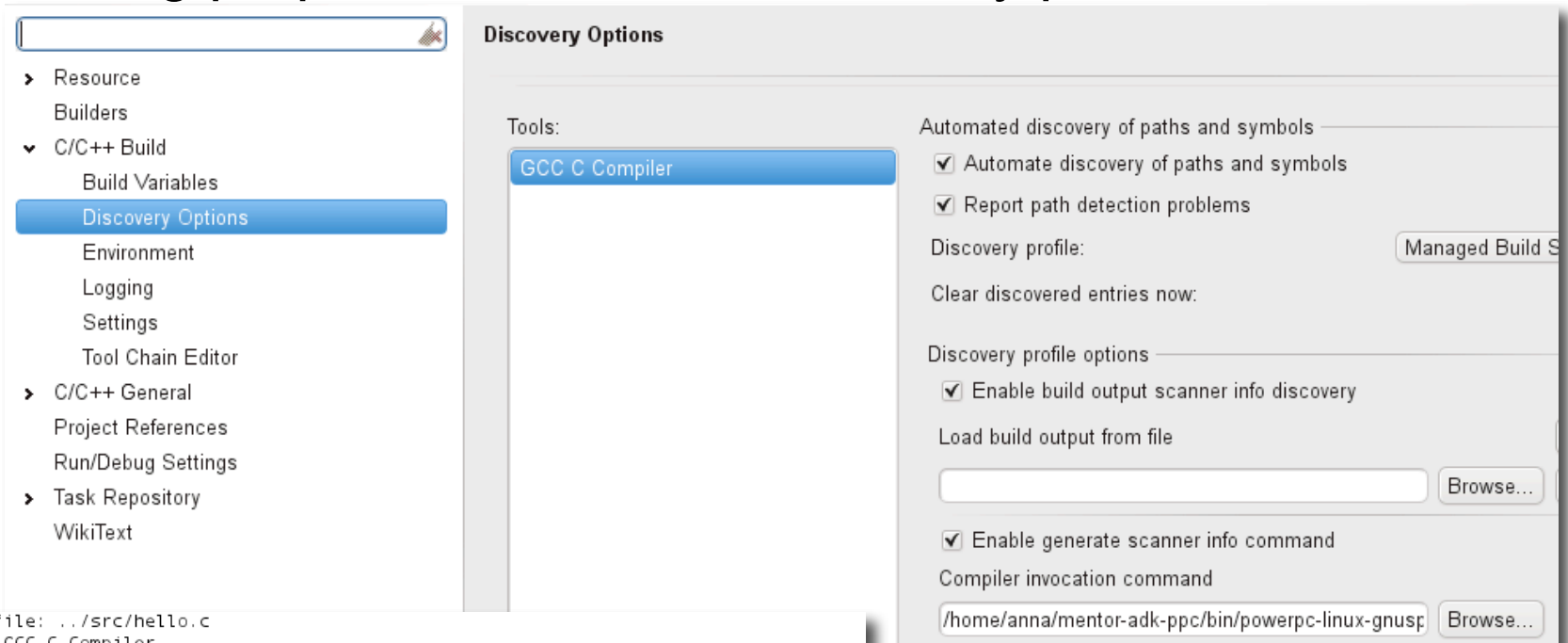
Redefining settings manually

- Has to be done for each configuration in each project in use
- Requires the following modifications of project properties:
 - changing gcc to your <arch>-gcc in Settings for each tool



Redefining settings manually - continued

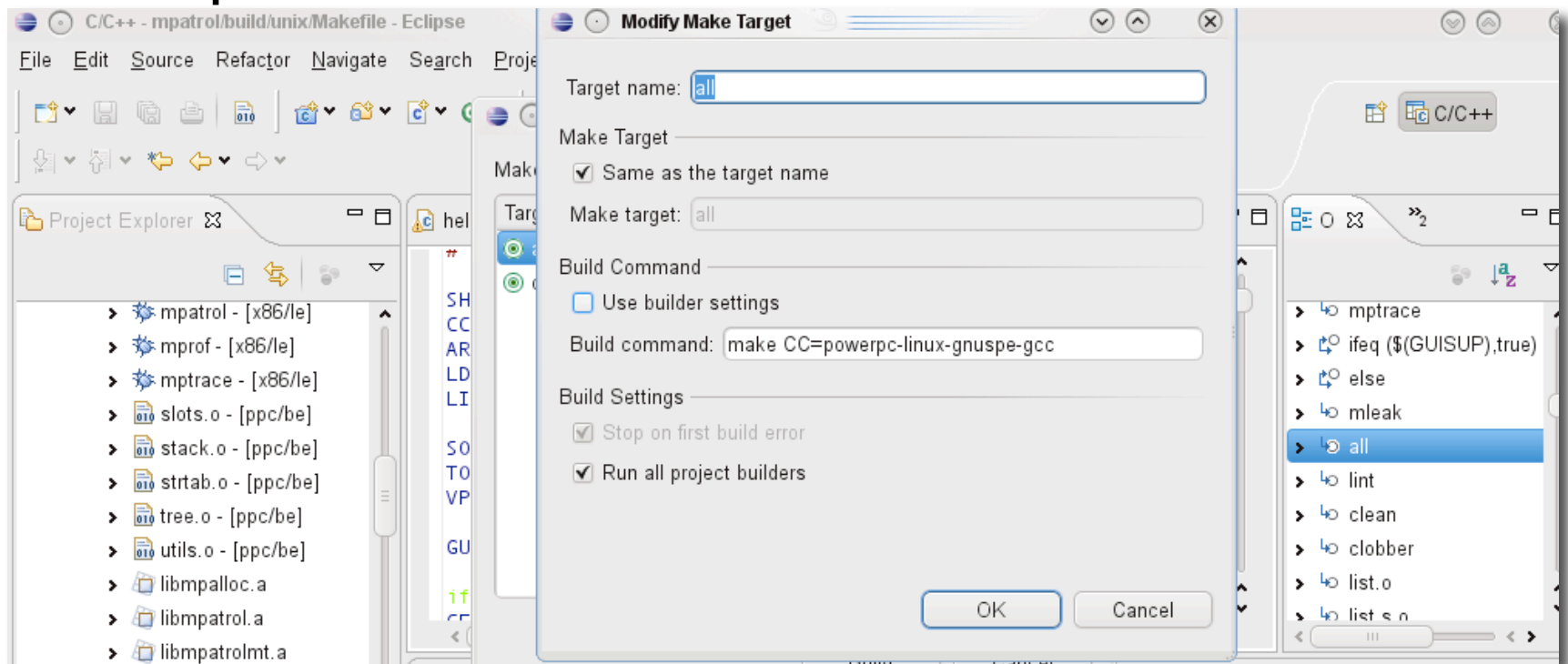
- Modifying PATH variable(adding your cross-toolchain path)
- Setting proper command for discovery profile



```
Building file: ../src/hello.c
Invoking: GCC C Compiler
powerpc-linux-gnusp-gcc -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"src/hello.d" -
MT"src/hello.d" -o"src/hello.o" "../src/hello.c"
../src/hello.c: In function 'main':
../src/hello.c:16: warning: suggest parentheses around assignment used as truth value
```

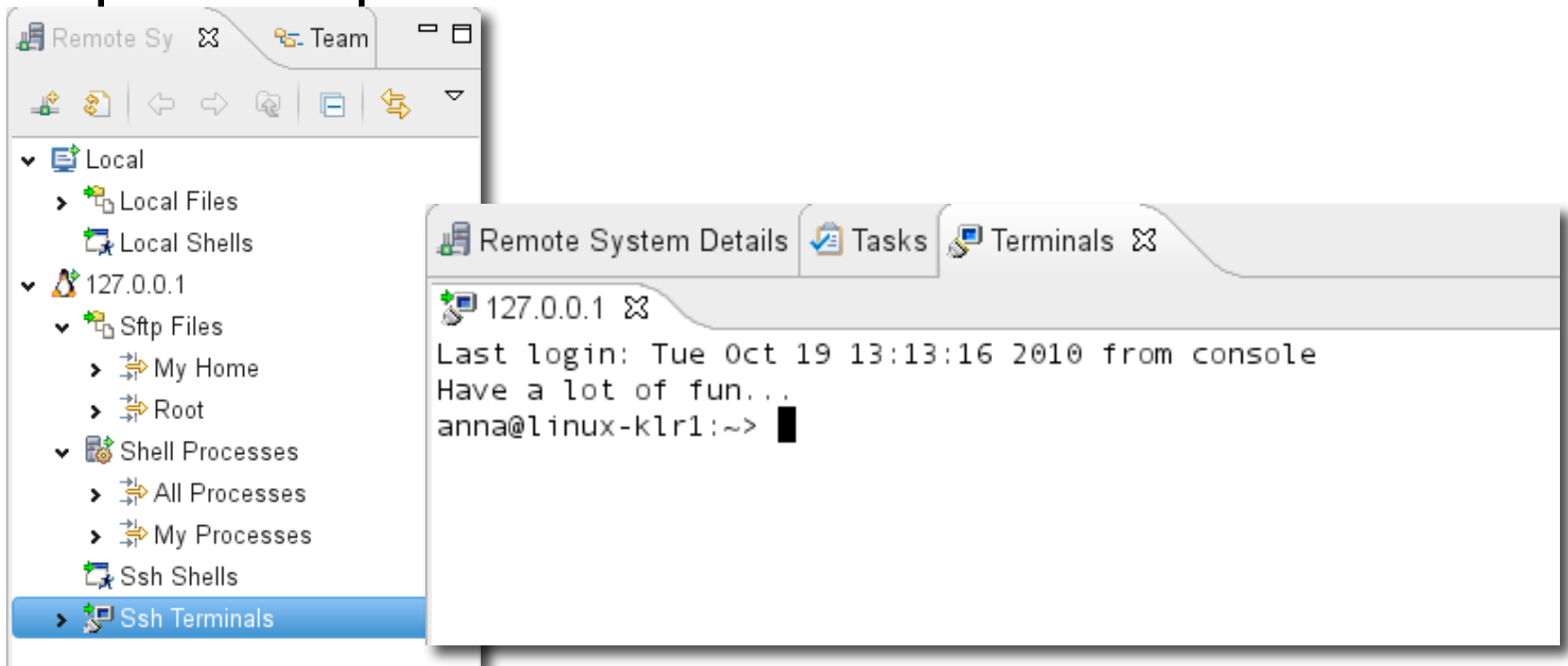
Makefiles

- Generated makefiles are not meant to be used outside CDT
- Better use “Makefile projects” for more control of your build process



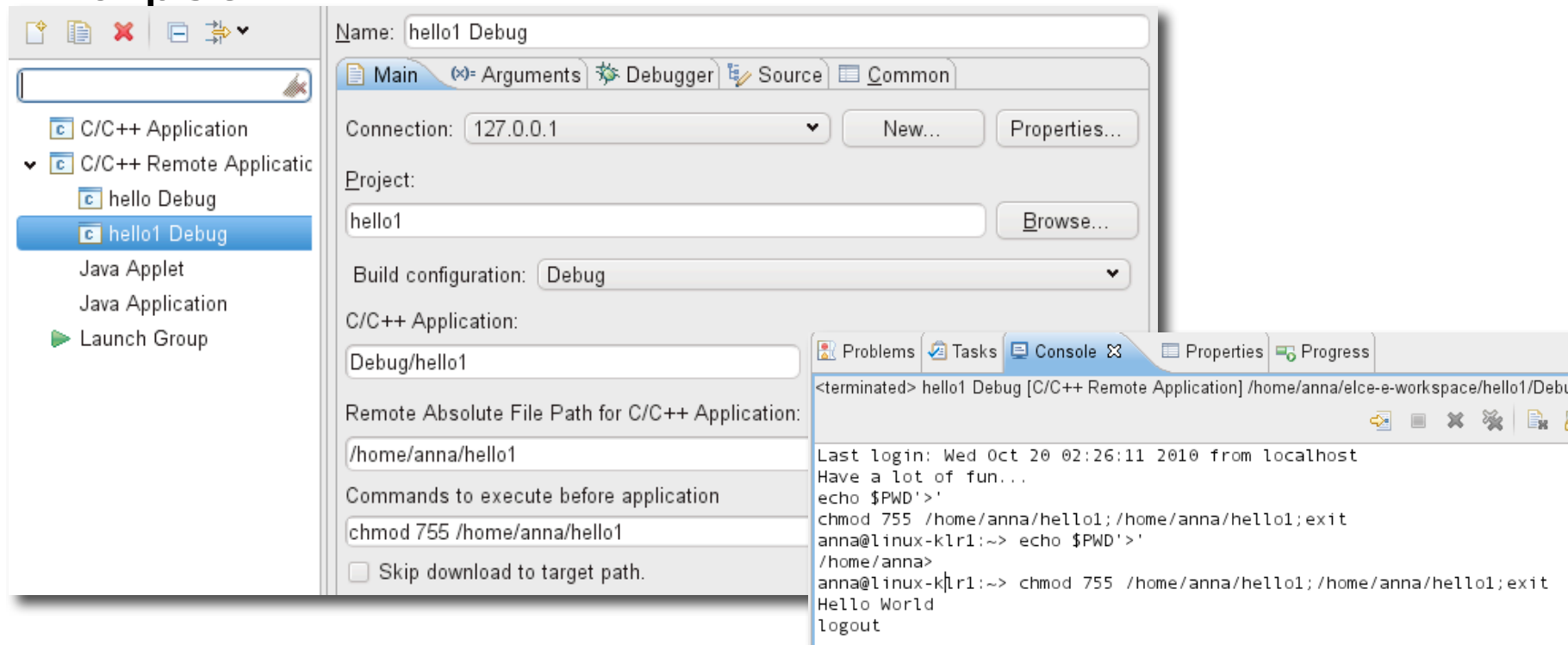
Working with remote systems

- If you have only serial, you can use the Target Management Terminal for board bring-up
- Once you have ethernet, you can use Remote System Explorer capabilities



Running your applications

- Use Remote Systems Explorer to copy your application on target and then terminal to launch it
- Use Remote Launch to launch your application from Eclipse



Debugging your applications

■ Variety of choices:

- remote debugger (launches gdbserver on remote side automatically, doesn't support attaching to a running process)
- “old”(CDI) CDT-GDB integration(you have to launch gdbserver manually, supports attach)
- “new”(DSF) CDT-GDB integration(you have to launch gdbserver manually, supports attach)

■ More coming soon:

- Eclipse Debugger for C/C++(agent on remote, Java implementation on host based on DSF)
- Agent based debugger(won't use DSF)

Debugging your applications-continued

The screenshot shows a debugger interface with the following components:

- Left Sidebar:** Displays the project structure. The selected item is `1 main() hello1.c:15 0x0804841d`.
- Main Editor:** Shows the source code of `hello1.c`. The current line is `puts("Hello World"); /* prints Hello World */`. Below it is `return EXIT_SUCCESS;`.
- Console:** Displays the output of the program. It shows a message about missing debuginfo for `/lib/libc.so.6` and a successful execution of `puts("Hello World")`. It also shows a breakpoint hit at `0x0804841d` in `main` at `../src/hello1.c:15`.
- Right Pane:** Contains two tabs: **Registers** and **Outline**. The **Registers** tab shows the values of `eax` (1) and `ecx` (-1077618060). The **Outline** tab shows the list of symbols: `stdio.h`, `stdlib.h`, and `main(void) : i`.

Comprehensive Solutions for

Android ▪ Nucleus ▪ Linux

Mobile & Beyond ▪ 2D/3D User Interfaces ▪ Multi-OS ▪ Networking

Linux measurement and diagnostic tools

- Relatively new in Eclipse

- Profiling

- oprofile: remote launching is not supported, only local
- valgrind: same

- Tracing

- doesn't have remote control
- requires building and installing Ittngtrace library prior to using the tool
- requires creating an Ittng project to import traces

Eclipse LTTng Integration

The screenshot displays the Eclipse LTTng Integration interface, which is divided into several panels:

- Control Flow**: A table listing processes and their execution details.
- Resources**: A tree view showing the hierarchy of resources, including CPUs, Event Types, and Modes.
- Statistics**: A table showing the number of events, CPU time, cumulative CPU time, and elapsed time for various resources.
- Events - test1**: A table showing the details of events, including timestamp, source, type, reference, and content.

Control Flow Table:

Process	Brand	PID	TGID	PPID	CPU	Birth sec	Birth nsec	TRAC
swapper		0	0	0	0	0	000000000	trace-
swapper		0	0	0	1	0	000000000	trace-
init		1	1	0	0	272	426910960	trace-
kthreadd		2	2	0	0	272	426915184	tra-
migration/0		3	3	2	0	272	426919008	tra-
ksoftirqd/0		4	4	2	0	272	426922848	tra-
migration/1		5	5	2	0	272	426926608	tra-
ksoftirqd/1		6	6	2	0	272	426930656	tra-
events/0		7	7	2	0	272	426934464	tra-

Statistics Table:

Level	Number of Events	CPU Time	Cumulative CPU Time	Elapsed Time
trace-p2020rdb	12641	239.014635136	486.195879687	486.074943942
CPUs				
0	6627	119.507575072	5668.895585506	436.063870957
Event Types				
Modes				
IRQ	441	0.000571299	0.008047242	0.000574803
MODE_UNKNOWN	761	0	0	0
SOFTIRQ	1263	0.003988823	0.05591565	0.003993975
SYSCALL	2195	119.496859648	6104.803010309	436.053996391
TRAP	1105	0.003055746	0.092483262	0.005305788
USER_MODE	862	0.003099556	0	0
1	6014	119.507060064	650.199621485	50.011072985
Event Types				
dev_xmit	2			
ioctl	2			

Events - test1 Table:

Timestamp	Source	Type	Reference	Content
272.426035585	Kernel Core	fs/1/pollfd	trace-p2020rdb	fd:55
272.426037505	Kernel Core	metadata/0/core_marker_id	trace-p2020rdb	alignment:0,int:4,size_t:4,name:vm_map,event_id:0
272.426041345	Kernel Core	fs/1/pollfd	trace-p2020rdb	fd:57
272.426043073	Kernel Core	fs/1/pollfd	trace-p2020rdb	fd:59
272.426043329	Kernel Core	metadata/0/core_marker_format	trace-p2020rdb	name:vm_map,format:pid %d start %lu end %lu fla
272.426044625	Kernel Core	fs/1/pollfd	trace-p2020rdb	fd:61
272.426046193	Kernel Core	fs/1/pollfd	trace-p2020rdb	fd:63
272.426046433	Kernel Core	metadata/0/core_marker_id	trace-p2020rdb	alignment:0,int:4,size_t:4,name:bio_complete,event
272.426047713	Kernel Core	fs/1/pollfd	trace-p2020rdb	fd:65

Comprehensive Solutions for

Android ▪ Nucleus ▪ Linux

Mobile & Beyond ▪ 2D/3D User Interfaces ▪ Multi-OS ▪ Networking

mentor
embedded

The result of our experiment

- Technology is there!
- But:
 - almost every piece requires non-trivial actions to make it work the way we need
 - some are just not possible to use for cross development
 - pieces do not integrate together very well
- Overall, the set of plugins and features we installed can hardly be called an IDE.

Questions?

Comprehensive Solutions for

Android ▪ **Nucleus** ▪ **Linux**

Mobile & Beyond ▪ 2D/3D User Interfaces ▪ Multi-OS ▪ Networking

Thank you!

Comprehensive Solutions for

Android ▪ **Nucleus** ▪ **Linux**

Mobile & Beyond ▪ 2D/3D User Interfaces ▪ Multi-OS ▪ Networking

mentor
embedded